Generative Adversarial Networks

Lecture 6

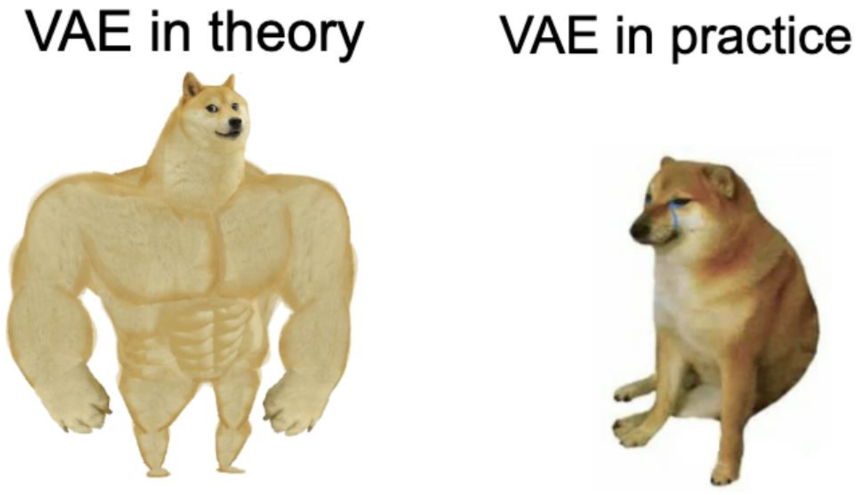
18-789

Recap

- VAEs maximize ELBO, a lower bound of the log-likelihood
  - $\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x) \parallel p(z))$
- Maximizing Gaussian $\log p_\theta(x|z)$ = Minimizing MSE/L2 loss
- Assume Gaussian $q_\phi(z|x)$ and $p(z)$ so we can compute their KL analytically
- Reparameterization trick to enable gradient backpropagation
- Autoencoder perspective, beta-VAE and VQVAE

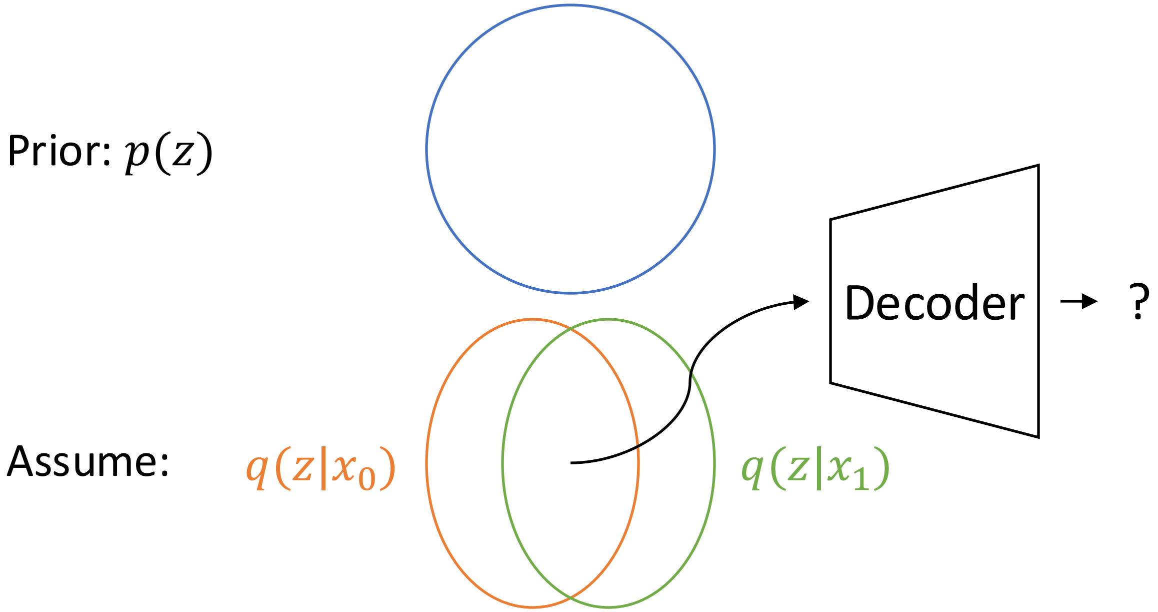
How do VAEs perform?

The output images are blurry!



Why are output images from VAEs blurry?

- Assume our dataset only has two samples: $\{x_0, x_1\}$.
- With optimized reconstruction loss, what would the decoder output if we sample from the origin?



$$\frac{x_0 + x_1}{2}$$

Why are output images from VAEs blurry?

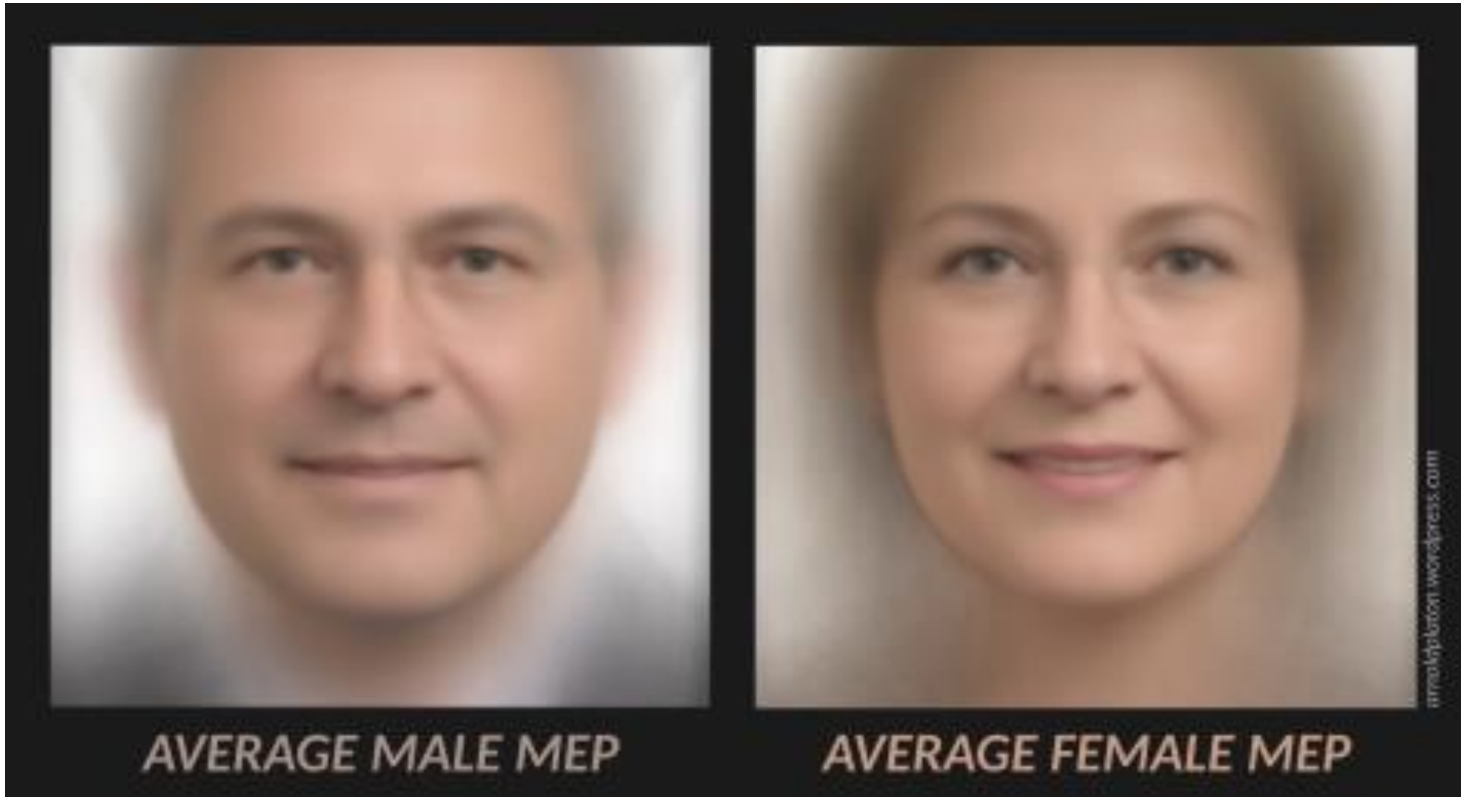
- The optimal decoder output given latent z' is a weighted average of samples in the training set $\{x_i\}$:

$$\mu_\theta(z') = \sum_i w_i x_i$$

where $w_i = \frac{q_\phi(z'|x_i)}{\sum_i q_\phi(z'|x_i)}$

The average of many images is blurry

- Average face of European parliament



Why are output images from VAEs blurry?

- The optimal decoder output given latent z' is a weighted combination of samples in the training set $\{x_i\}$:

$$\mu_\theta(z') = \sum_i w_i x_i$$

where $w_i = \frac{q_\phi(z'|x_i)}{\sum_i q_\phi(z'|x_i)}$

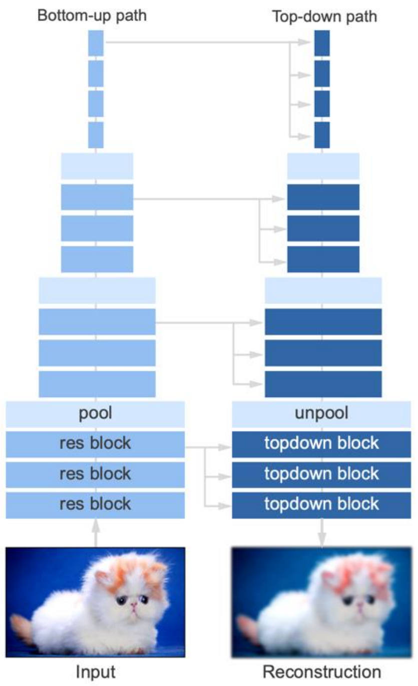
- Another answer: VAE outputs are blurry because of Gaussian assumptions
  - Gaussian likelihood -> The optimal decoder output is a weighted average of training samples.
  - Gaussian posterior + prior -> There is always overlap between posterior distributions, so weights are never one-hot.

VAE: The Curse of Blurriness

- Blurriness is a fundamental problem of VAEs that can't be easily solved by scaling up

256x256

1024x1024



Very Deep (72 layers) +
Hierarchical Latent Space

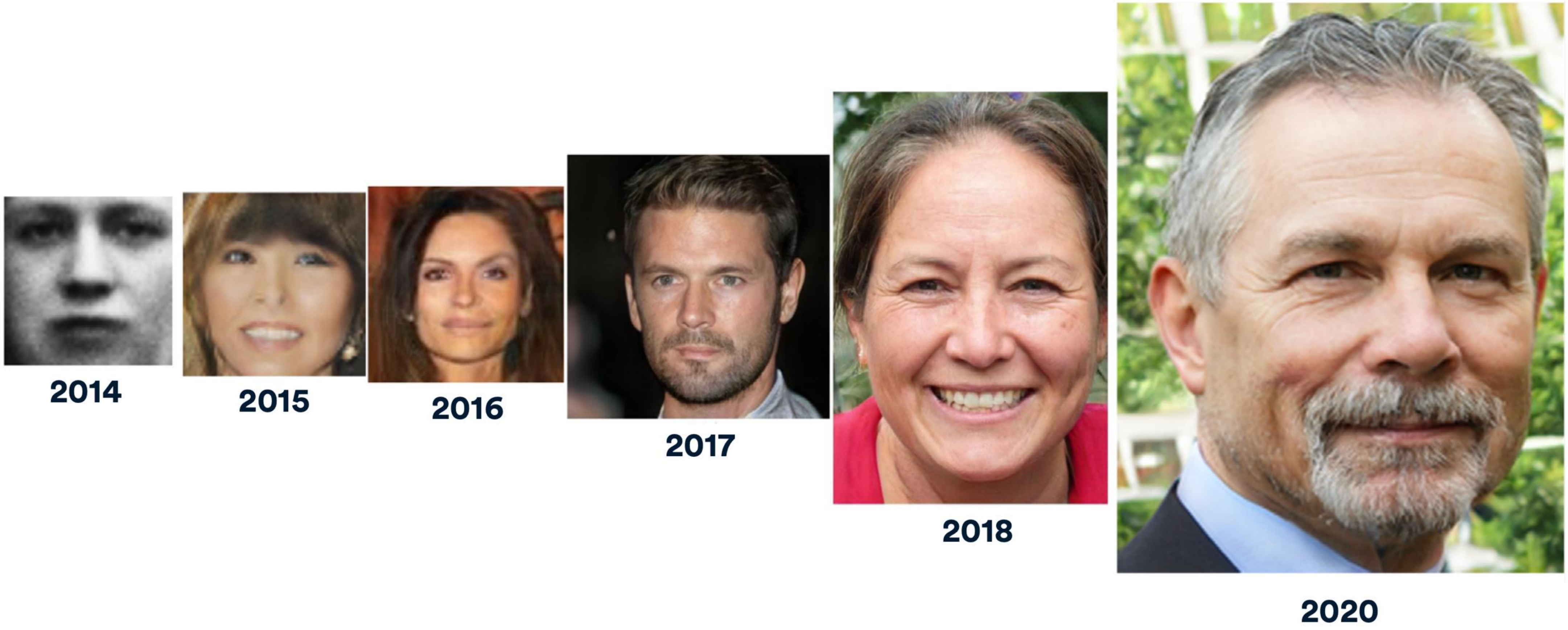
Child, Very deep vaes generalize autoregressive models and can outperform them on images, ICLR 2020

Non-Gaussian Decoder?

- $\min_{\theta,\phi} -\mathbb{E}_{z\sim q_\phi(z|x)}[\log \underbrace{p_\theta(x|z)}_{\text{PixelCNN}}] + KL(q_\phi(z|x) \parallel p(z))$
- Can we use an autoregressive decoder (e.g. PixelCNN)?
- Posterior Collapse!
  - Encoder ignores x ($q_\phi(z|x) \equiv p(z)$ regardless of x)
  - Decoder ignores z ($p_\theta(x|z) \equiv p_\theta(x)$ regardless of z)
- Degenerate to an autoregressive model…

Chen et al., "Variational lossy autoencoder", ICLR 2017

The Evolution of GANs

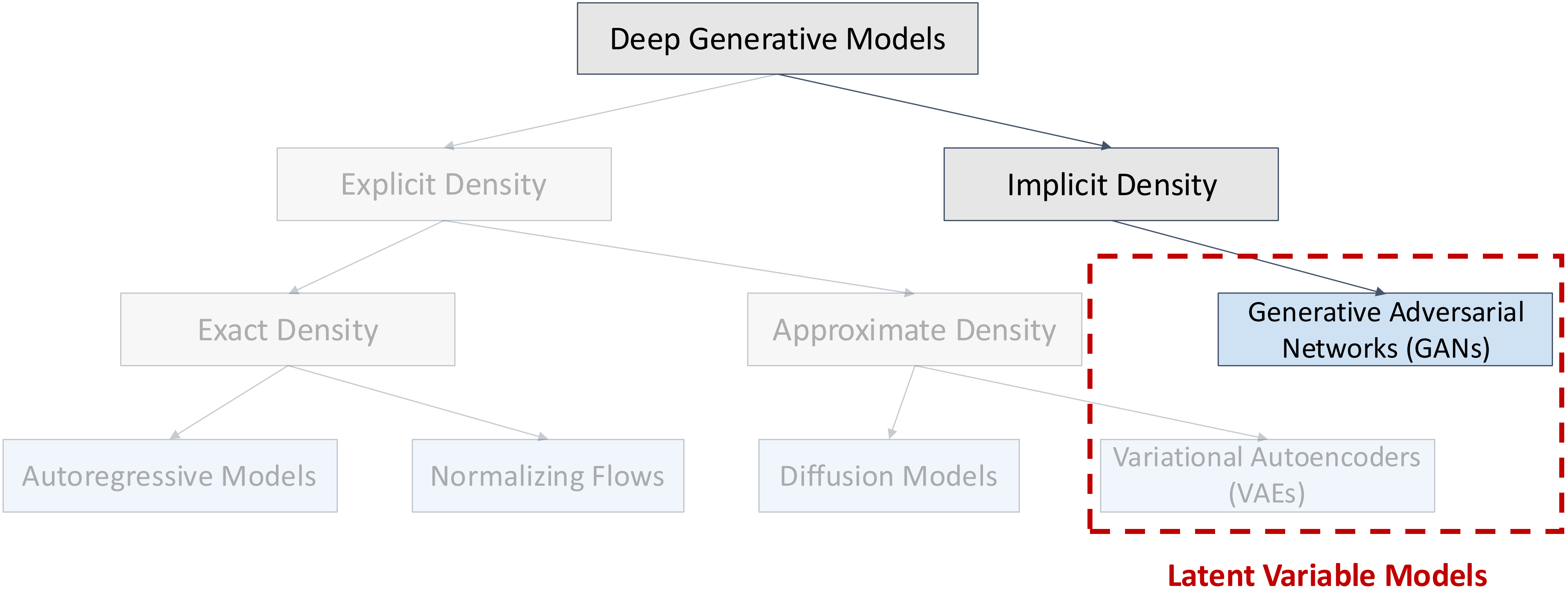


Goodfellow et al., 2014; Radford et al., 2016; Liu & Tuzel, 2016; Karras et al., 2018; Karras et al., 2019; Goodfellow, 2019; Karras et al., 2020; AI Index, 2021

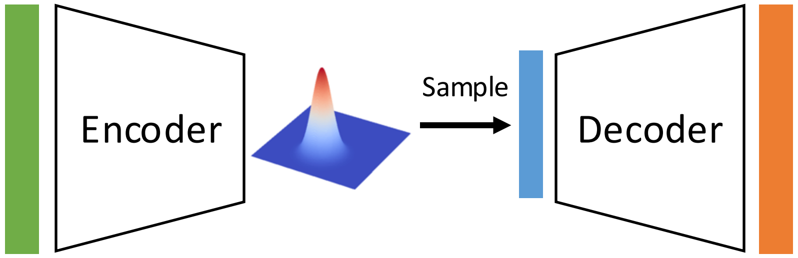
GANs are still widely used today, especially when combined with Diffusion

Lin et al., “Diffusion Adversarial Post-Training for One-Step Video Generation”

Generative Adversarial Networks



Recap: Variational Autoencoders



1. How to model the joint distribution of high-dimensional data?
   - $p_\theta(x) = \int_z p(z) p_\theta(x|z) \mathrm{dz}$, where z is lower-dimensional
   - $p(z)$ and $p_\theta(x|z)$ are simple independent Gaussian distributions
     - $p(z) = N(0, I)$
     - $p(x|z) = N(\mu_\theta(z), \sigma I)$

2. How to optimize your model?

Maximizing ELBO (lower bound of likelihood) + Reparameterization

Had to make Gaussian assumptions so that ELBO is tractable to compute

As a result: sample quality not good (blurry)

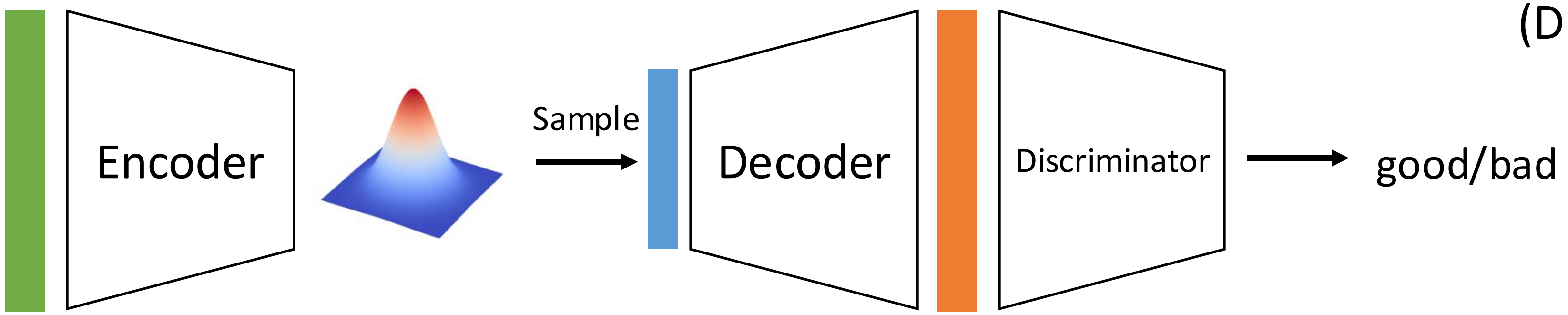
From VAEs to GANs

1. How to model the joint distribution of high-dimensional data?
   - $p_\theta(x) = \int_z p(z) p_\theta(x|z) \mathrm{d}z$, where z is lower-dimensional
   - $p(z)$ and $p_\theta(x|z)$ are simple independent Gaussian distributions
     - $p(z) = N(0, I)$
     - $p(x|z) = N(\mu_\theta(z), \sigma)$

2. How to optimize your model?

   Minimizing ~~KL Divergence~~ some distance learned by a neural network

   (Discriminator)



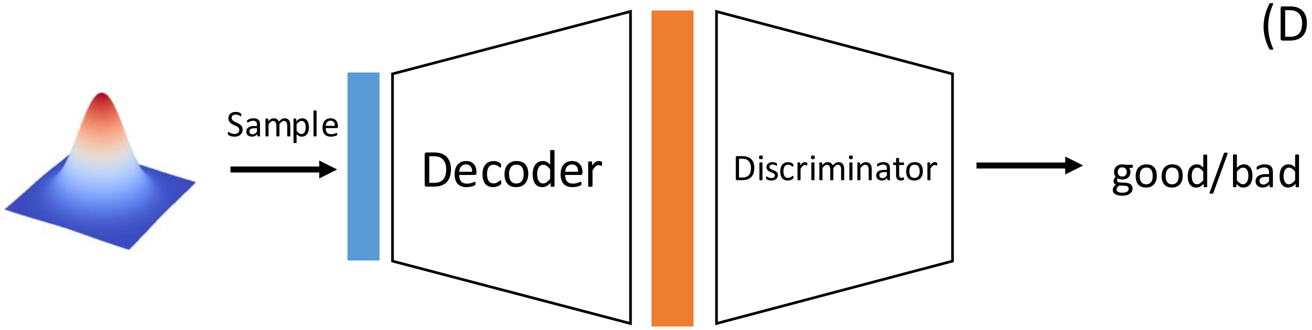
From VAEs to GANs

1. How to model the joint distribution of high-dimensional data?
   - $p_\theta(x) = \int_z p(z) p_\theta(x|z) \mathrm{dz}$, where z is lower-dimensional
   - $p(z)$ and $p_\theta(x|z)$ are simple independent Gaussian distributions
     - $p(z) = N(0, I)$
     - $p(x|z) = N(\mu_\theta(z), \sigma), \sigma \to 0 \Longrightarrow x = \mu_\theta(z)$

2. How to optimize your model?

   Minimizing ~~KL Divergence~~ some distance learned by a neural network

(Discriminator)



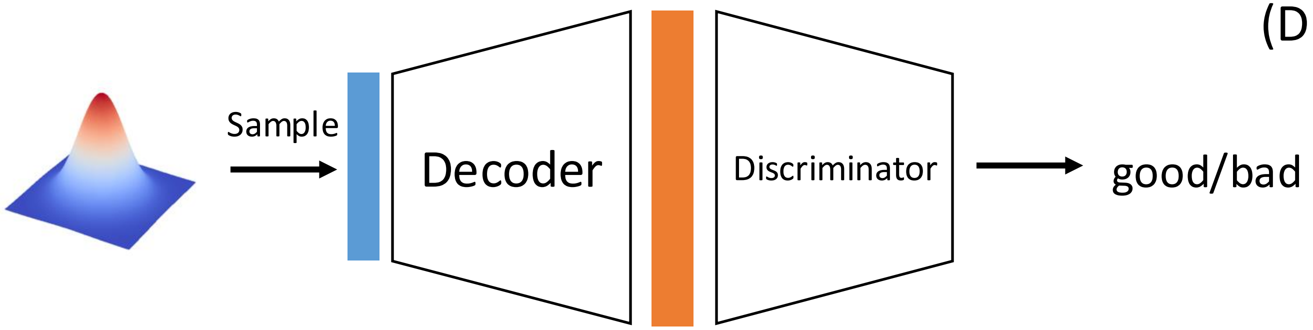
From VAEs to GANs

1. How to model the joint distribution of high-dimensional data?
    - (Implicit) $p_\theta(x)$: $x = G_\theta(z), z \sim p(z)$, where z is lower-dimensional
    - $p(z) = N(0, I)$ is an independent Gaussian (or any simple distribution that is easy to sample from)

2. How to optimize your model?

   Minimizing ~~KL Divergence~~ some distance learned by a neural network

   (Discriminator)



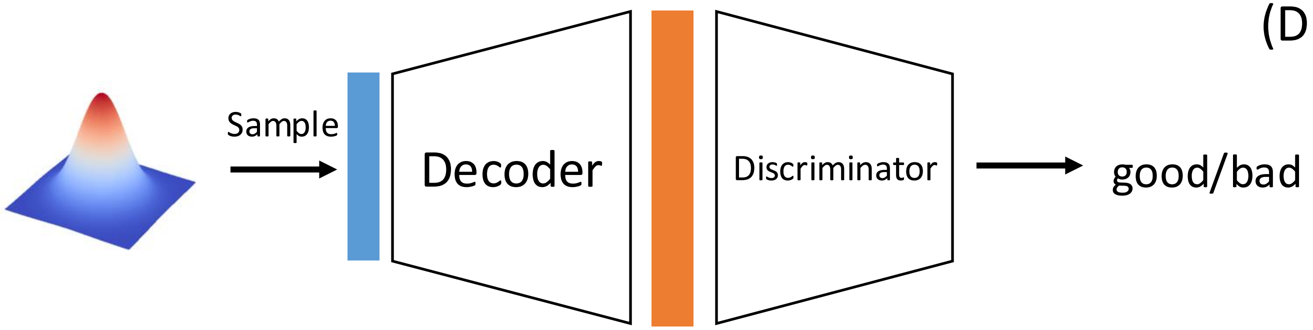
From VAEs to GANs

1. How to model the joint distribution of high-dimensional data?
   - (Implicit) $p_\theta(x): x = G_\theta(z), z \sim p(z)$, where z is lower-dimensional
   - $p(z) = N(0, I)$ is an independent Gaussian (or any simple distribution that is easy to sample from)

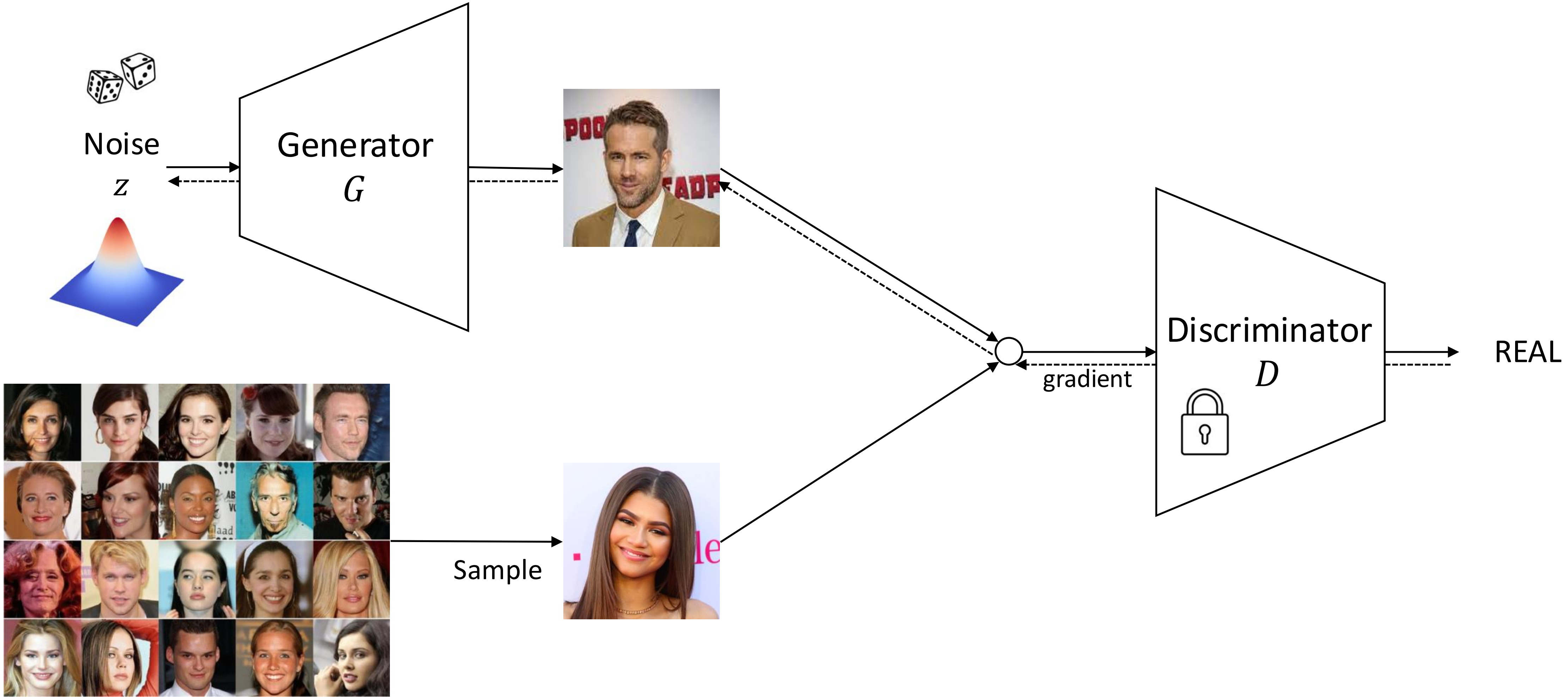
2. How to optimize your model?

Minimizing ~~KL Divergence~~ some distance learned by a neural network

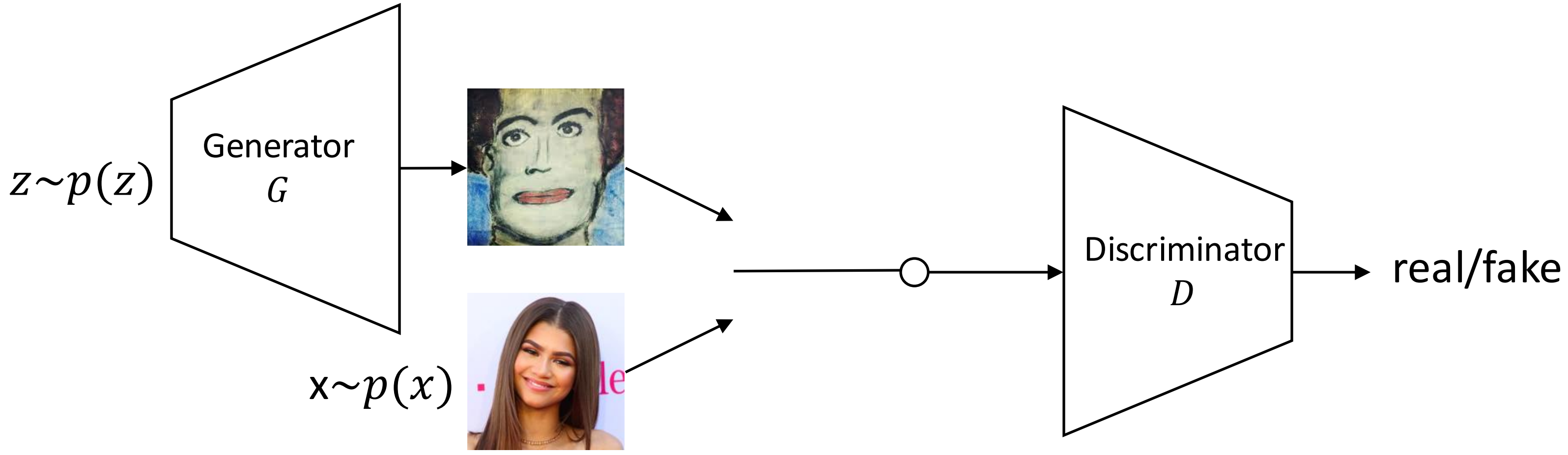
(Discriminator)



Intuition



GAN Objective



Inner optimization:

Generated sample

$$\min_{G} \max_{D} \boxed{E_{x\sim p(x)}[\log D(x)]} + \boxed{E_{z\sim p(z)}[\log(1 - D(G(z)))]}$$

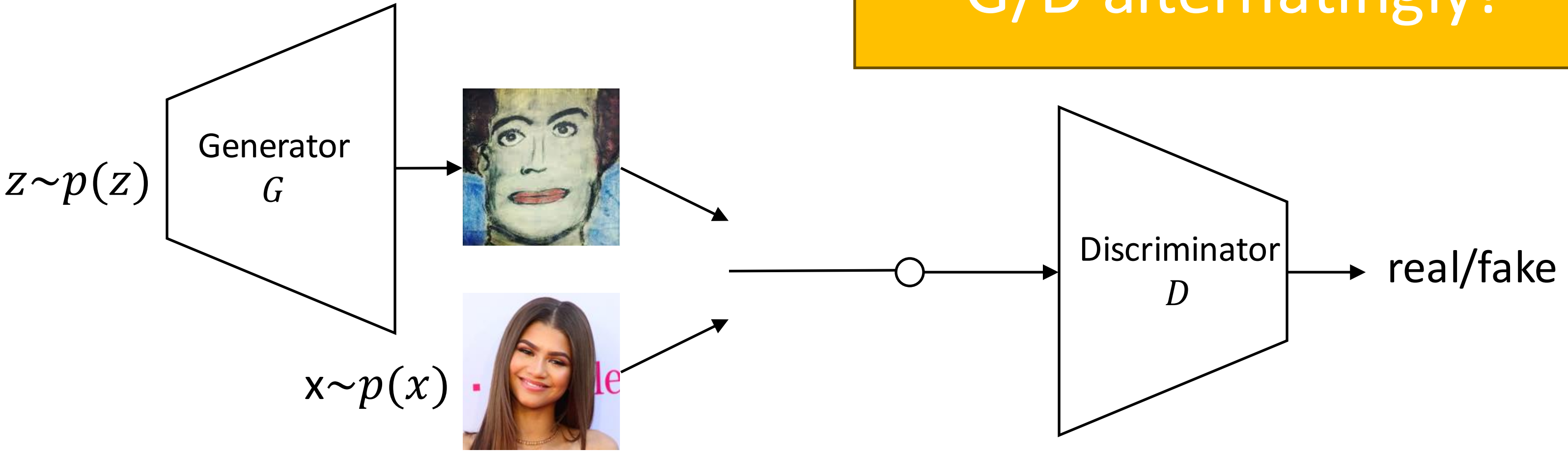
Maximize discriminator output for real data

Minimize discriminator output for generated data

Training discriminator with binary classification loss

GAN Objective

In practice, we update G/D alternatingly!



Outer optimization:

Generated sample

$$\min_G \max_D E_{x\sim p(x)}[\log D(x)] + \boxed{E_{z\sim p(z)}[\log(1-D(G(z)))]}$$

Maximize discriminator output for generated data

Generator tries to fool the discriminator!

Discriminator estimates the density ratio

For a fixed generator G (with parameter θ), the optimal discriminator is

$$D^*(x) = \frac{p(x)}{p(x) + p_\theta(x)}$$

where $p(x)$ and $p_\theta(x)$ are data and model distribution.

$$\min_G \max_D E_{x\sim p(x)}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))]$$

Generator can learn true data distribution

The global optimum of the GAN objective is achieved if and only if

$$p_\theta(x) = p(x)$$

$$\begin{aligned}
\mathcal{L}(G) &= E_{x\sim p(x)}[\log D^*(x)] + E_{x\sim p_\theta(x)}[\log(1 - D^*(G(z)))] \\
&= E_{x\sim p(x)}\left[\log \frac{p(x)}{p(x)+p_\theta(x)}\right] + E_{x\sim p_\theta(x)}\left[\log(1 - \frac{p(x)}{p(x)+p_\theta(x)})\right] + \log(4) - \log(4) \\
&= E_{x\sim p(x)}\left[\log \frac{2p(x)}{p(x)+p_\theta(x)}\right] + E_{x\sim p_\theta(x)}\left[\log \frac{2p_\theta(x)}{p(x)+p_\theta(x)})\right] - \log(4) \\
&= \boxed{KL\left(p(x)\middle|\middle| \frac{p(x)+p_\theta(x)}{2}\right) + KL\left(p_\theta(x)\middle|\middle| \frac{p(x)+p_\theta(x)}{2}\right)} - \log(4)
\end{aligned}$$

Jensen-Shannon divergence: 2JSD$(p(x)|p_\theta(x))$

How do GANs perform?

GANs can perform really well *when it works*.

StyleGAN, Karras et al., 2019

Failure scenario: **Mode Collapse**

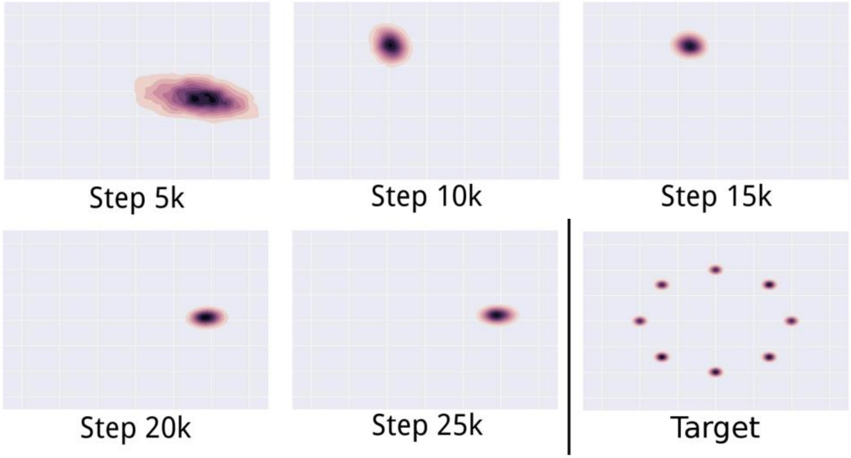
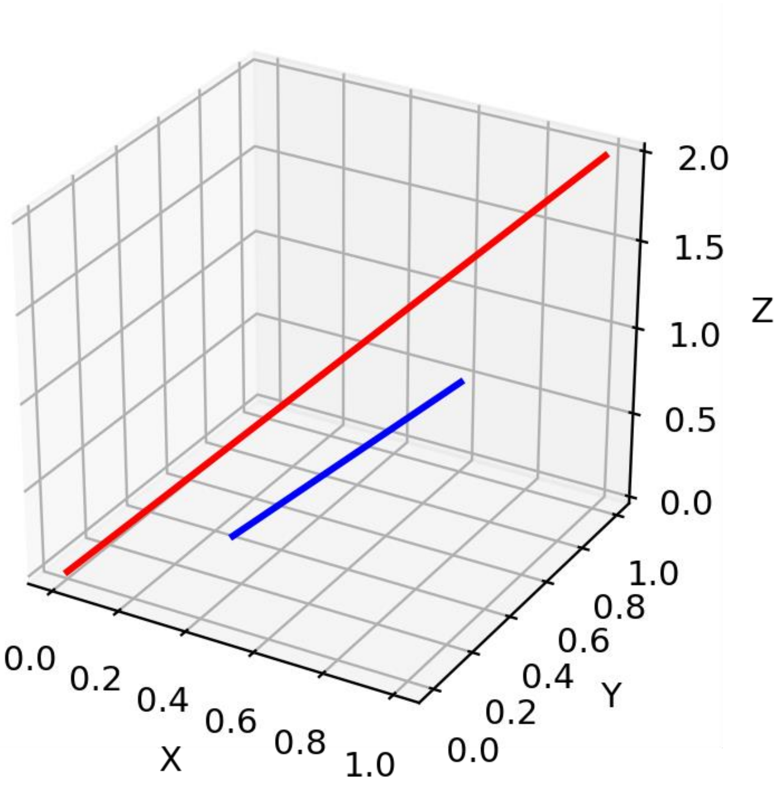
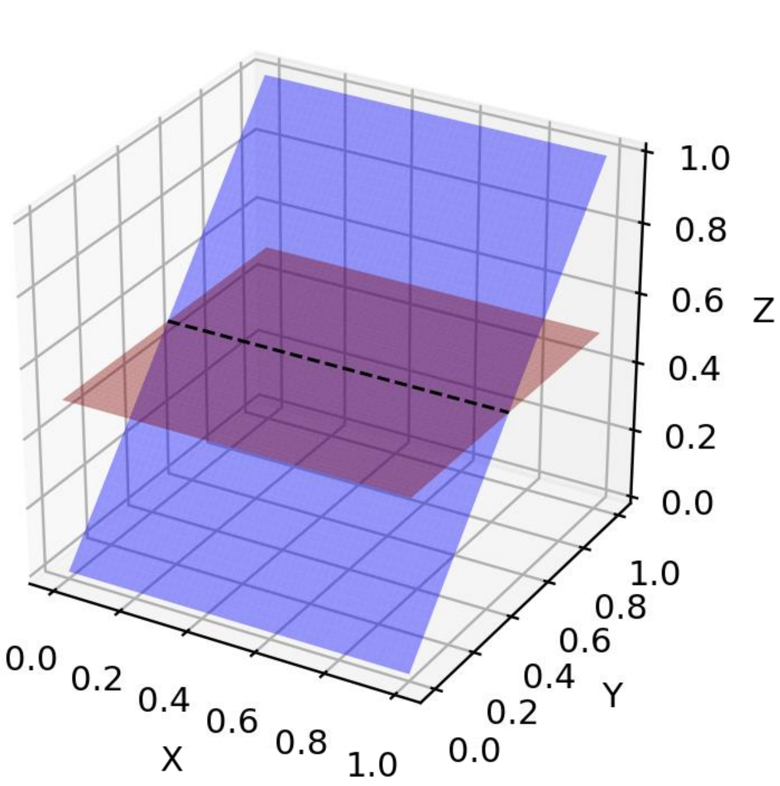


Image source: Arjovsky et al. 2017

Manifold hypothesis

- High-dimensional data sets in the real world (e.g., images) actually lie along low-dimensional manifolds.



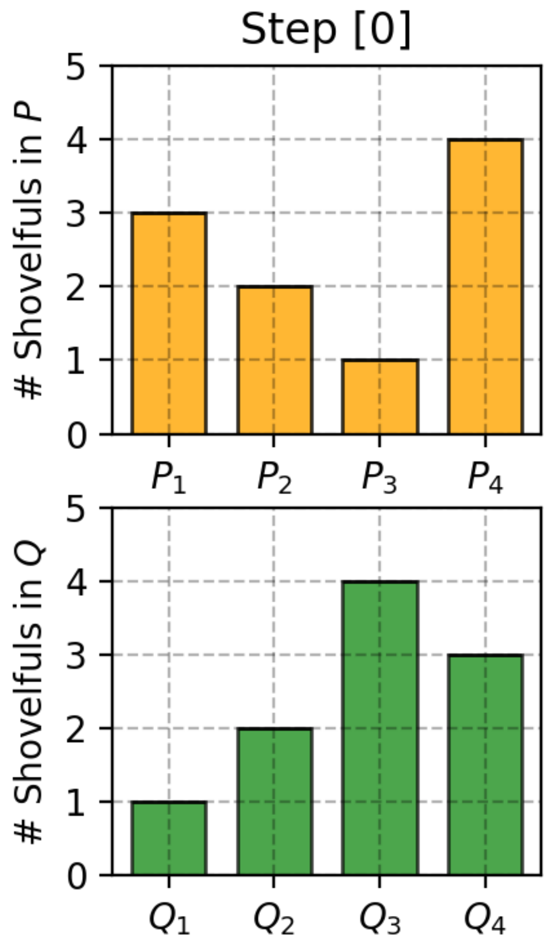


A hypothesis on the cause of mode collapse

- $\text{JSD}(p|p_\theta) = \frac{1}{2} KL\left(p(x) || \frac{p(x)+p_\theta(x)}{2}\right) + \frac{1}{2} KL\left(p_\theta(x) || \frac{p(x)+p_\theta(x)}{2}\right)$
- If p and p_θ have completely different supports:
  - $\text{JSD}(p|p_\theta) = \log 2$
  - No matter what our parameters are!
  - There’s no gradient!

- Claim: GAN training is unstable because it’s optimizing JSD
- Solution: Let’s optimize another distance

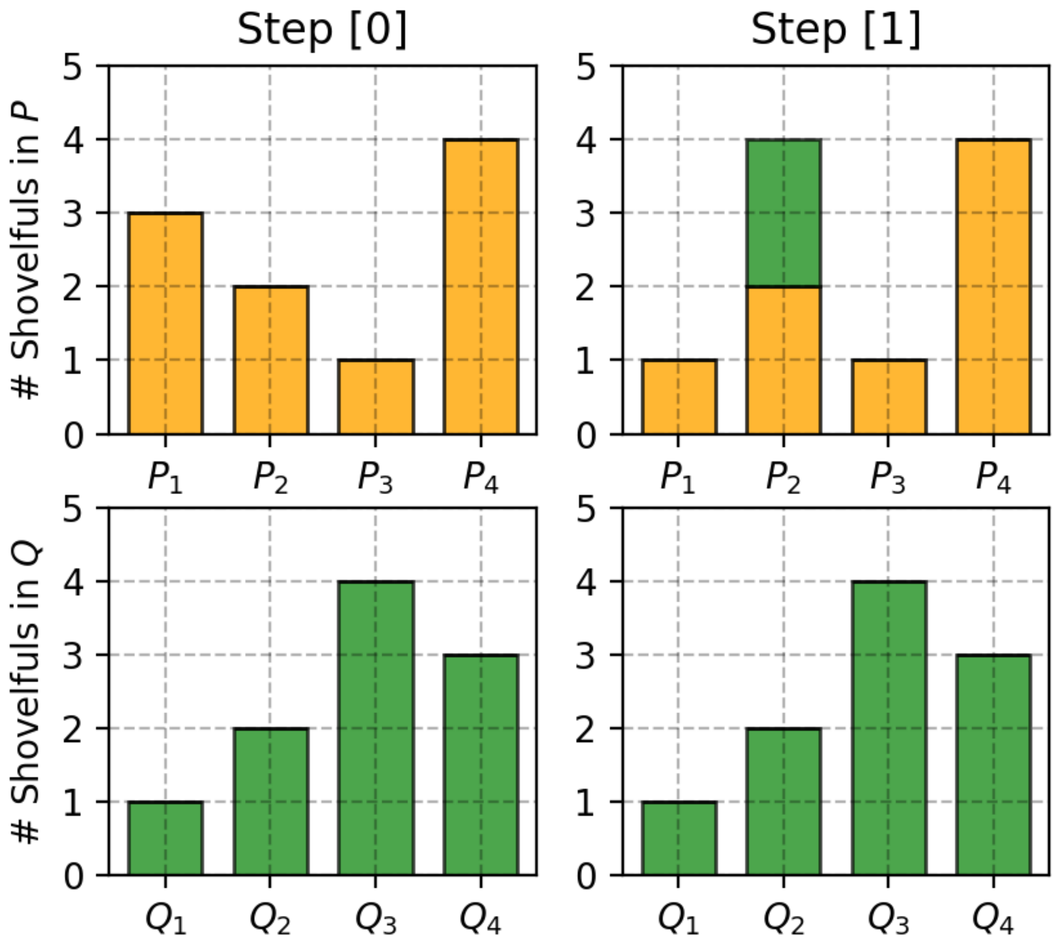
Wasserstein distance



What's the "minimum work" we need to move distribution P to Q?

Source: Lilian Weng, "From GAN to WGAN"

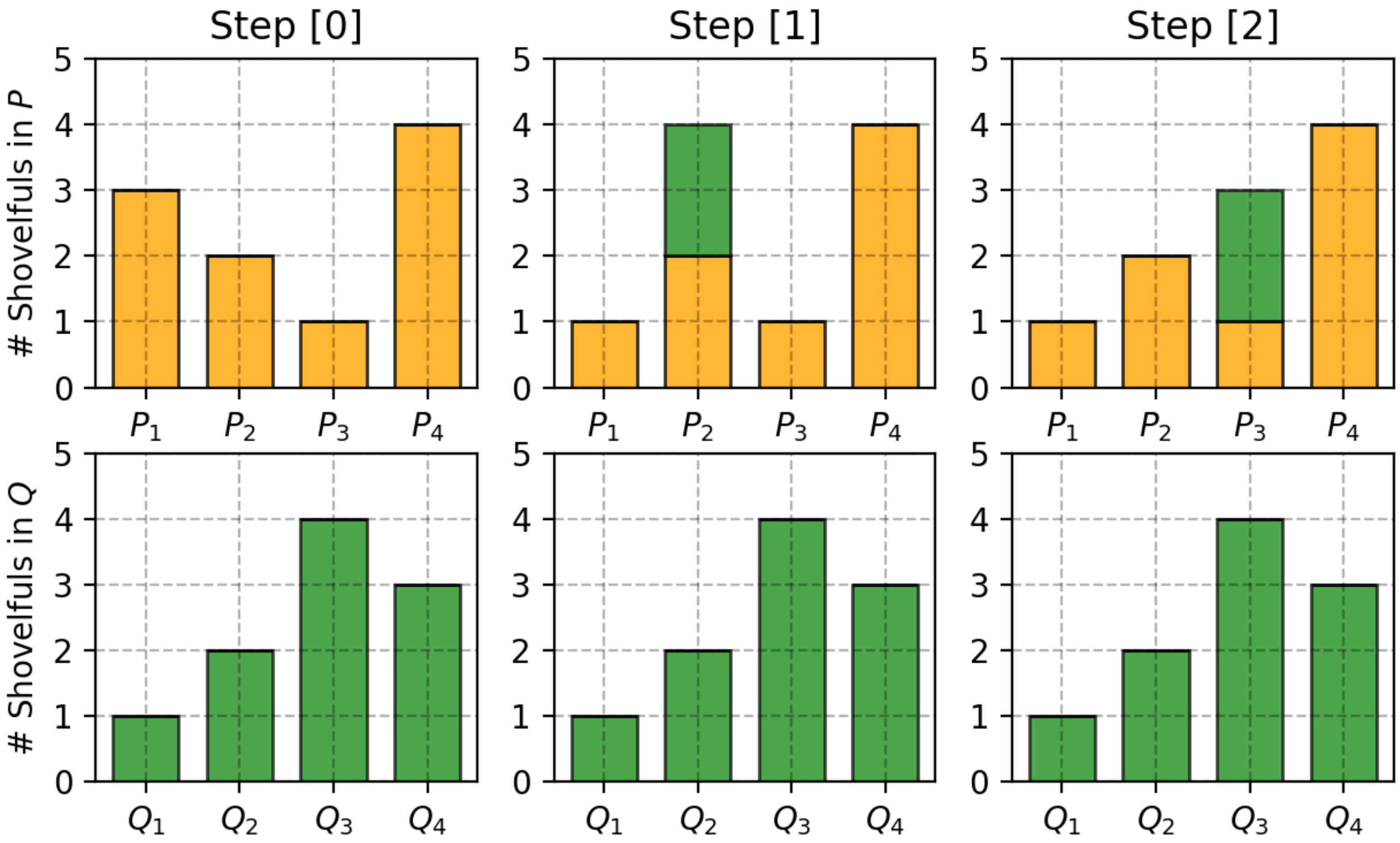
Wasserstein distance



2

Source: Lilian Weng, “From GAN to WGAN”

Wasserstein distance



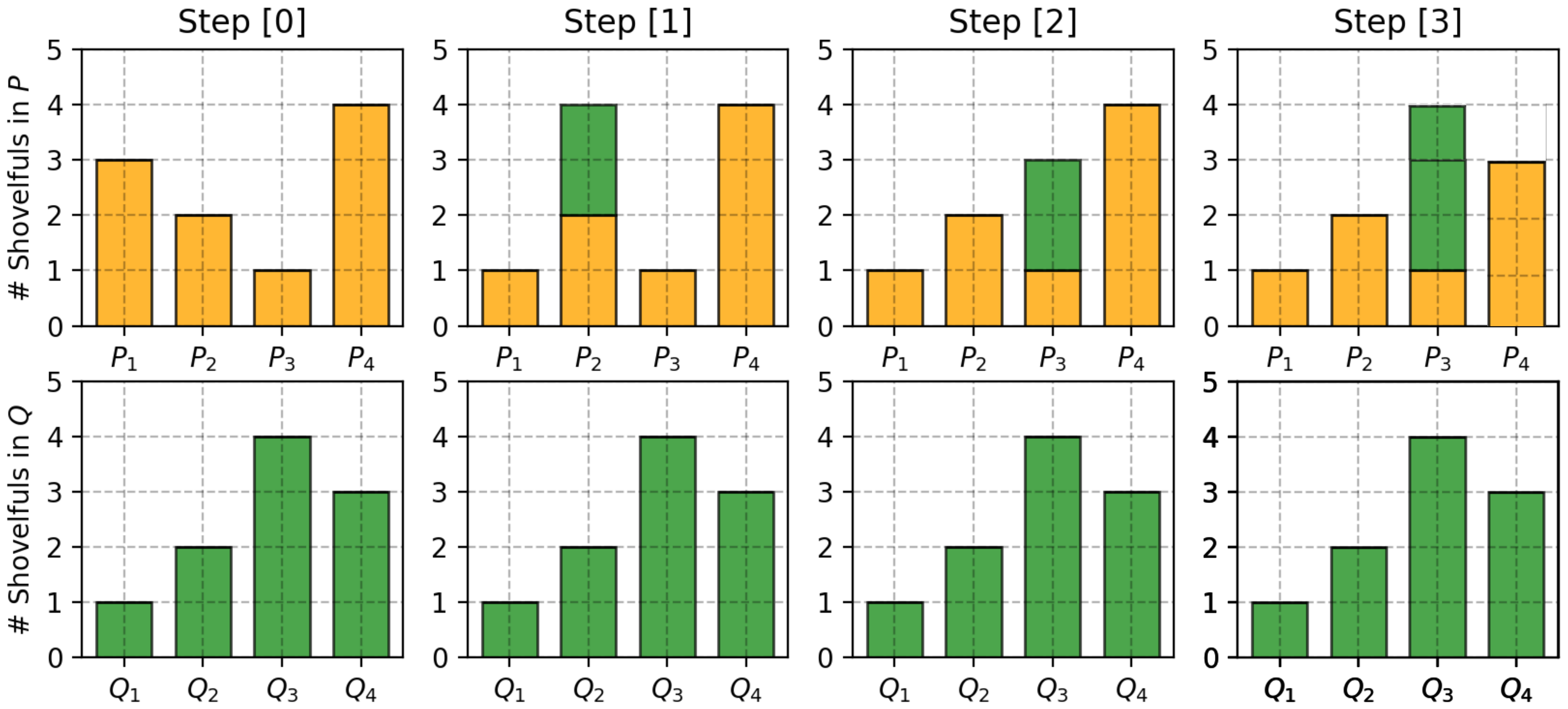
2+2

Source: Lilian Weng, "From GAN to WGAN"

Wasserstein distance

Transport plan:

1	0	2	0
0	2	0	0
0	0	1	0
0	0	1	3



2+2+1=5

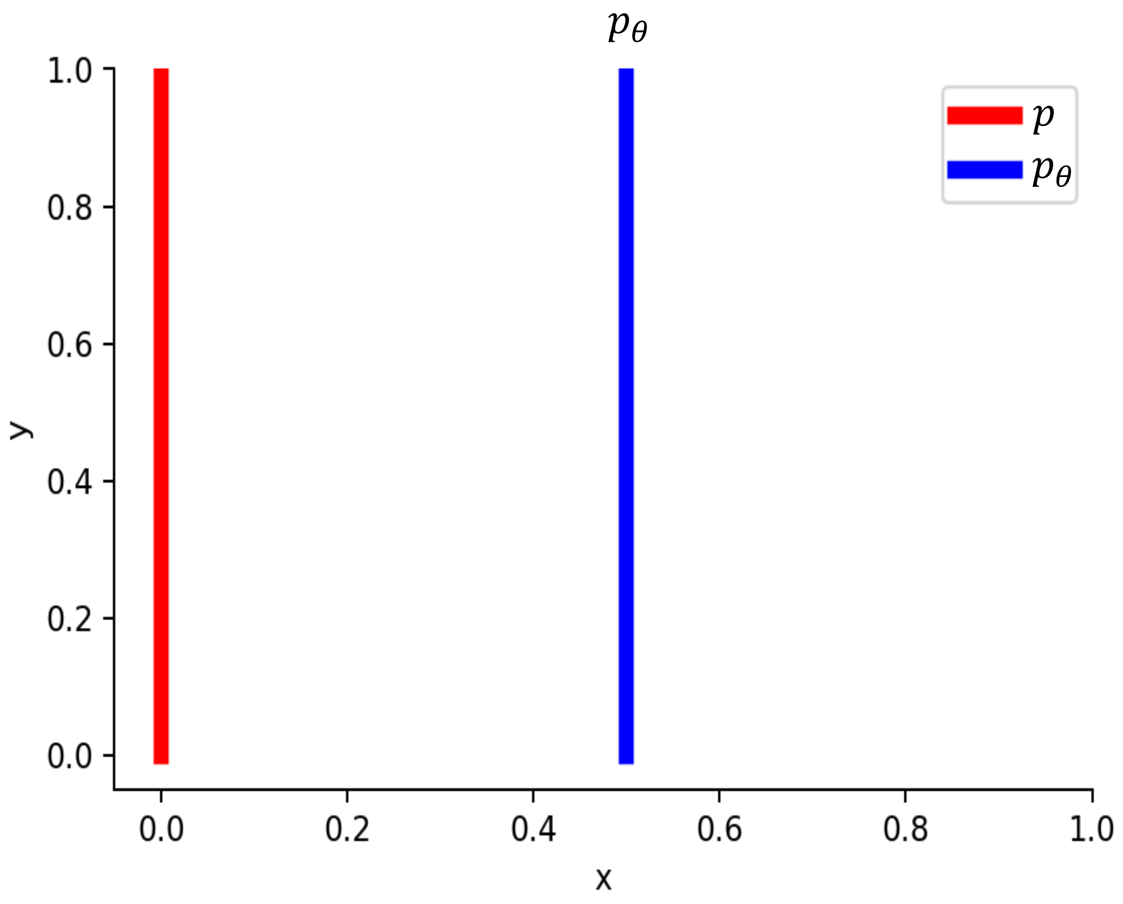
Source: Lilian Weng, “From GAN to WGAN”

Wasserstein distance

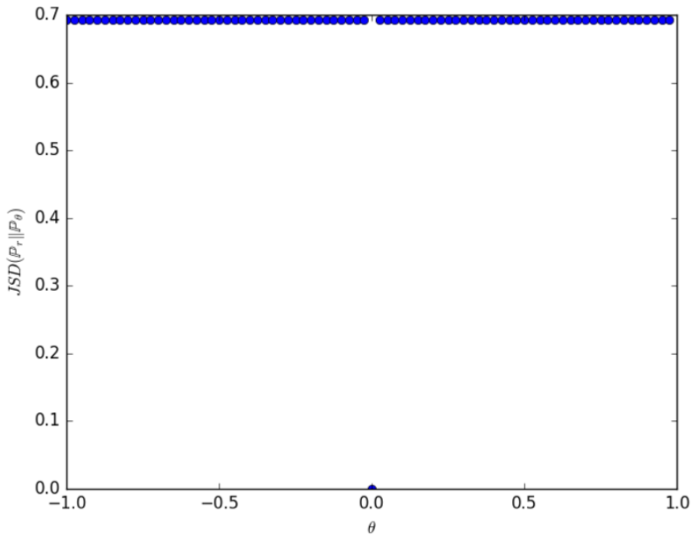
- Wasserstein distance: for $\Pi(p, p_\theta)$ defined as all possible joint probability distributions between p and p_θ

$$W(p, p_\theta) = \inf_{\gamma \in \Pi(p, p_\theta)} E_{(x,y)\sim\gamma}[\ ||x - y||\ ]$$

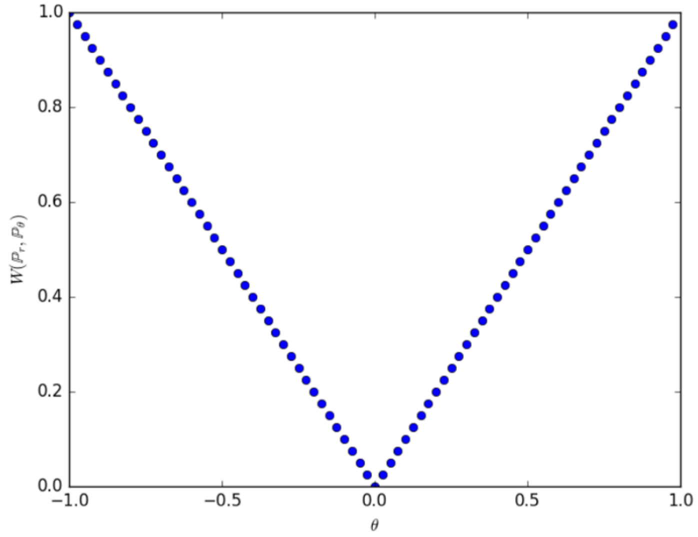
Why Wasserstein might be better than JSD?



Gradient of JSD



Gradient of WD



Estimating the Wasserstein Distance

$$W(p, p_\theta) = \inf_{\gamma \in \Pi(p, p_\theta)} E_{(x,y)\sim\gamma}[\ ||x - y||\ ]$$

- Kantorovich-Rubinstein duality:

$$= \sup_{||f||_L \leq 1} E_{x\sim p}[f(x)] - E_{x\sim p_\theta}[f(x)]$$

- $||f||_L \leq 1 : f$ is 1-Lipschitz

$$\frac{|f(x) - f(y)|}{|x - y|} \leq 1, \forall x, y$$

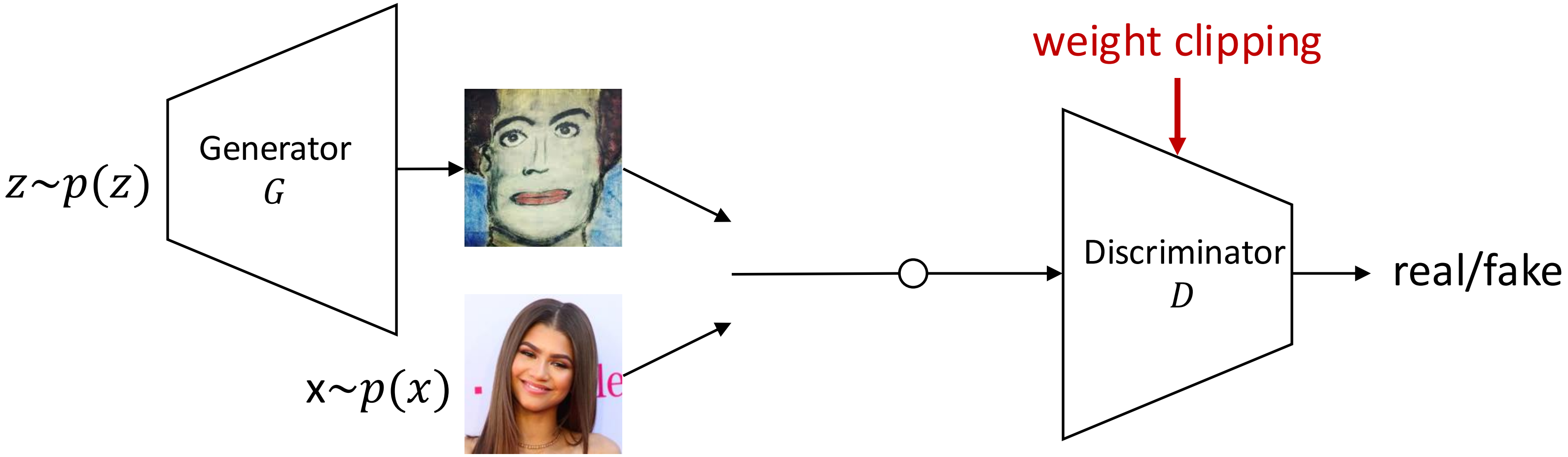
Bounded gradient!

Wasserstein GAN in practice

- Use the discriminator as "f" :

$$\max_{||D||_L \leq 1} E_{x\sim p}[D(x)] - E_{x\sim p_\theta}[D(x)]$$

- Removed log compared with the original objective
- Use weight clipping to ensure Lipschitz condition



Despite the nice theory…

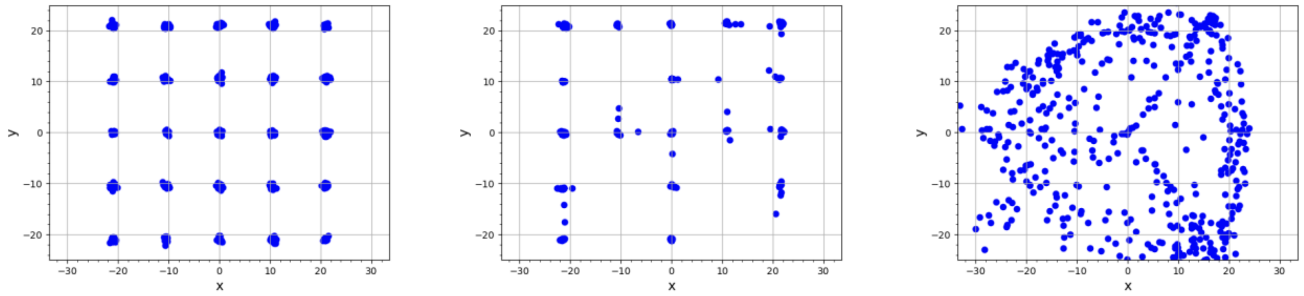
Stack Overflow
https://stackoverflow.com › questions › training-stabilit...

Training stability of Wasserstein GANs

The problem is that GANs not having a unified objective functions ... **wGAN** could be faster due to having morestable training procedures ...

2 answers · Top answer: You can check Inception Score and Frechet Inception Distance for now. And a…

The loss of my **WGAN** slumps to the negative infinity within just ... Apr 21, 2023
WGAN-GP Large Oscillating Loss - tensorflow - Stack Overflow Dec 30, 2019
WGAN loss diverges - Stack Overflow Mar 27, 2020



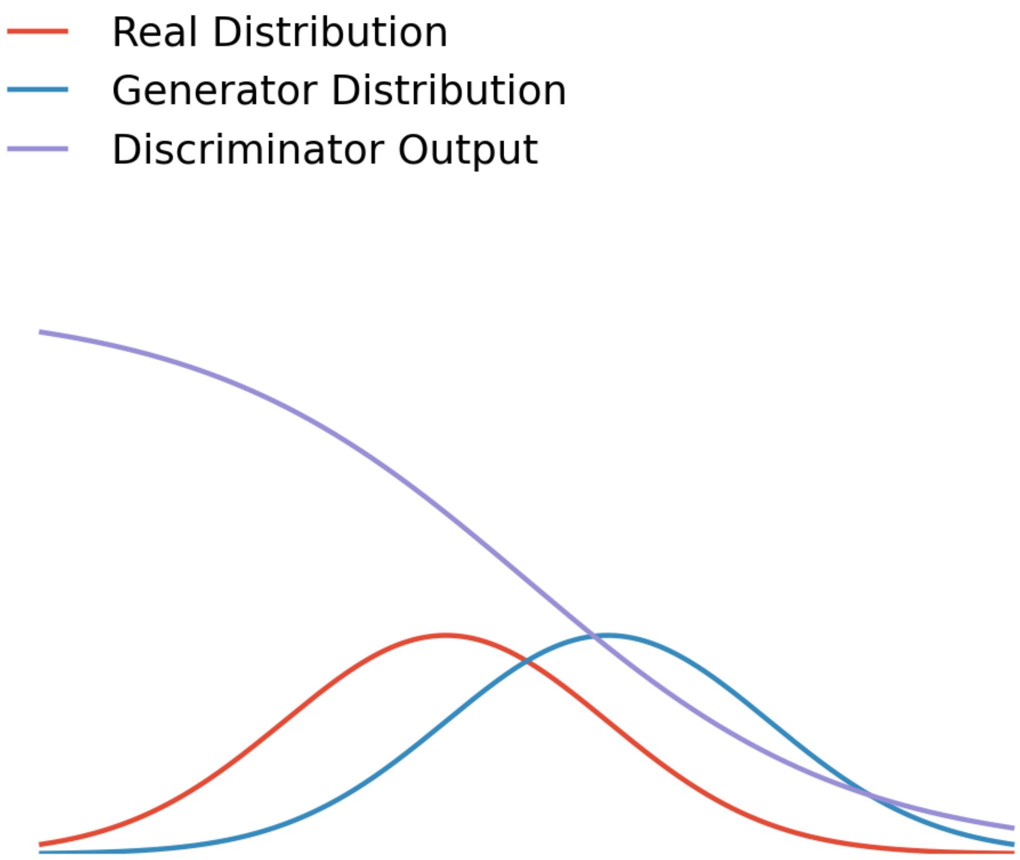
(a) True data (b) GAN (d) WGAN

GANs are optimizing [...] divergence. **Or do they?**

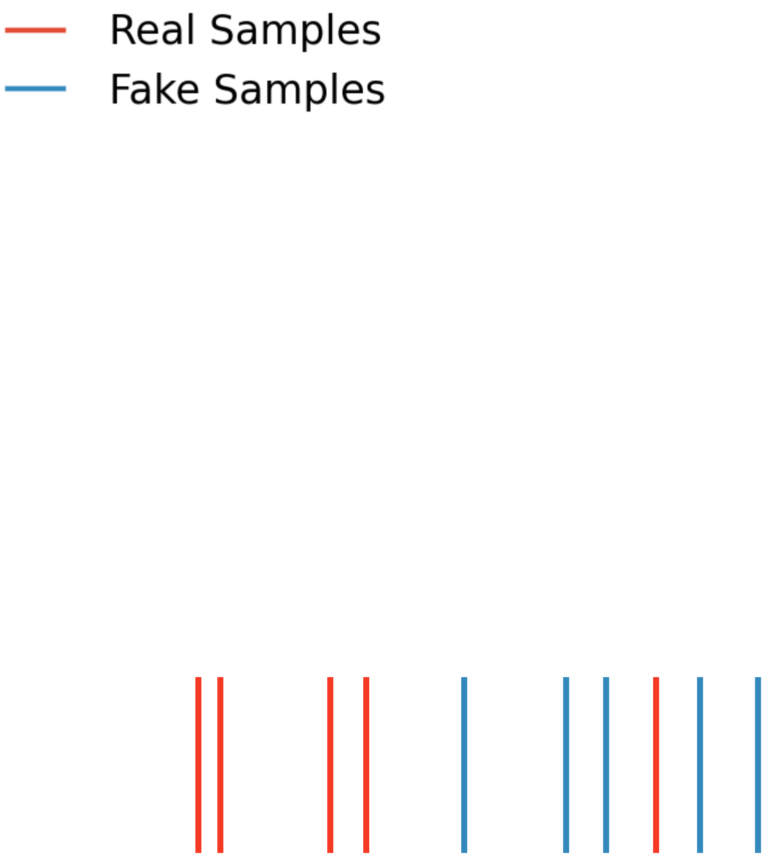
- *Theoretically*, the generator is optimizing some divergence (JSD/Wasserstein distance) *if we train the discriminator to optimal*.
- In practice, we are *never* going to train the discriminator to optimal.
  - Impractical
  - Overfitting
- In practice, GANs can work well in situations where the divergence minimization view predicts they would fail.
- It's more helpful to think the discriminator as some learned "neural network divergence" rather than a fixed mathematical divergence.

Fedus et al., "Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step"
Qing et al., "Do GAN Loss Functions Really Matter?"

Culprit of GAN Training Instability

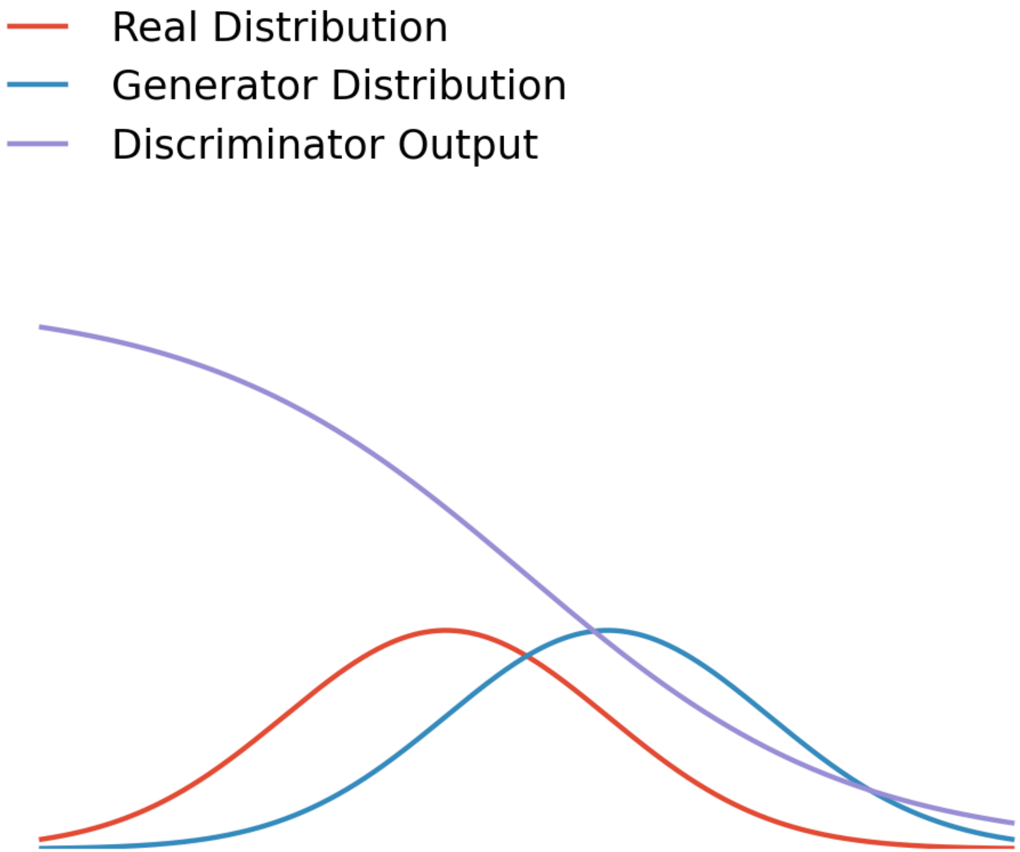


In theory

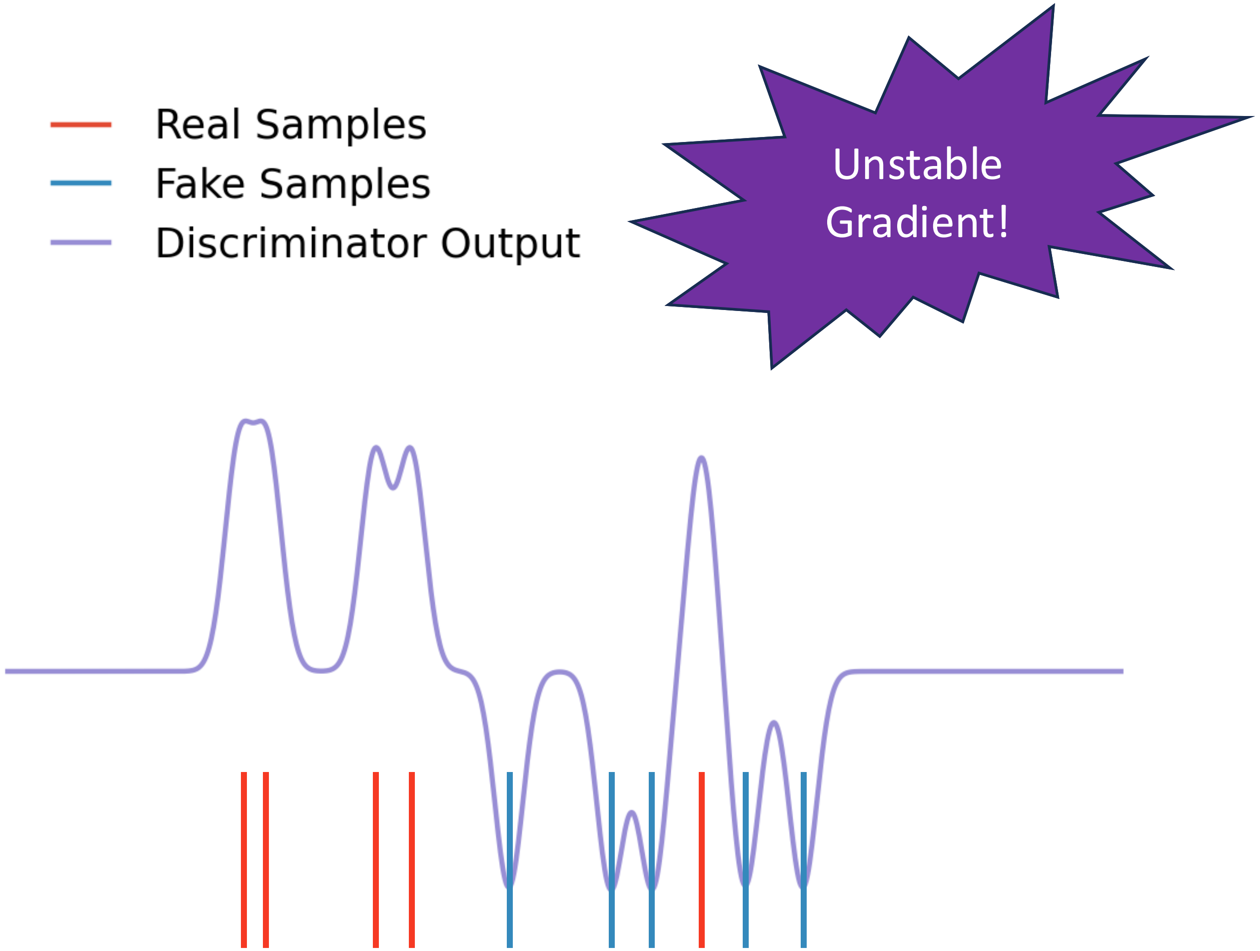


In practice

Culprit of GAN Training Instability



In theory



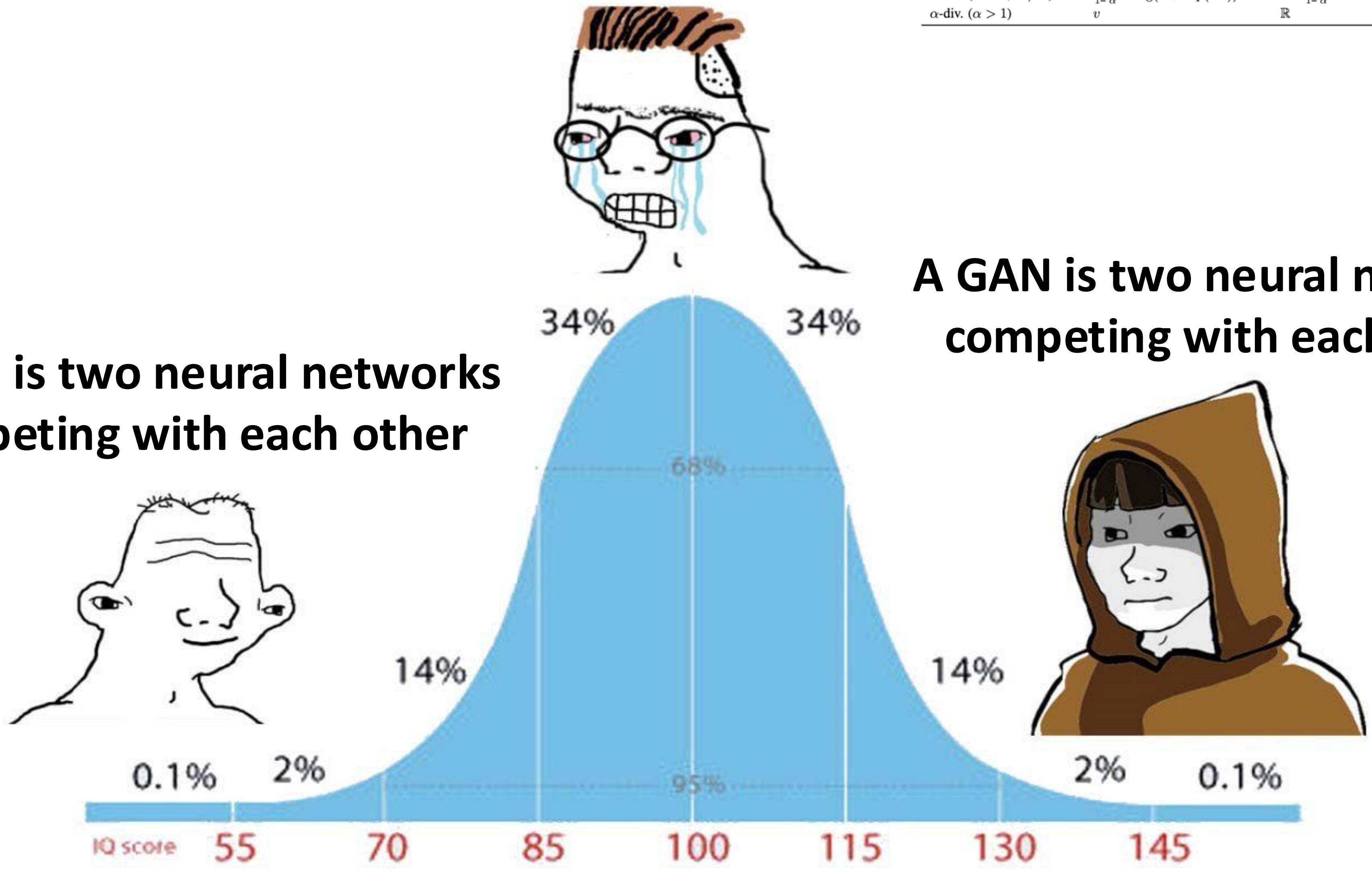
In practice

NOOO! A GAN is optimizing Jensen-Shannon Divergence/Wasserstein Distance/f-divergence

Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Total variation	$\frac{1}{2}\tanh(v)$	$-\frac{1}{2} \leq t \leq \frac{1}{2}$	t	0
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(v)$	$\mathbb{R}_-$	$-1-\log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2+t$	0
Neyman χ^2	$1-\exp(v)$	$t<1$	$2-2\sqrt{1-t}$	0
Squared Hellinger	$1-\exp(v)$	$t<1$	$\frac{t}{1-t}$	0
Jeffrey	v	$\mathbb{R}$	$W(e^{1-t})+\frac{1}{W(e^{1-t})}+t-2$	0
Jensen-Shannon	$\log(2)-\log(1+\exp(-v))$	$t<\log(2)$	$-\log(2-\exp(t))$	0
Jensen-Shannon-weighted	$-\pi\log\pi-\log(1+\exp(-v))$	$t<-\pi\log\pi$	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$
α-div. ($\alpha<1, \alpha\neq 0$)	$\frac{1}{1-\alpha}-\log(1+\exp(-v))$	$t<\frac{1}{1-\alpha}$	$\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}}-\frac{1}{\alpha}$	0
α-div. ($\alpha>1$)	v	$\mathbb{R}$	$\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}}-\frac{1}{\alpha}$	0

A GAN is two neural networks competing with each other

A GAN is two neural networks competing with each other



Next: Student Presentation
Improving the Stability of Training GANs

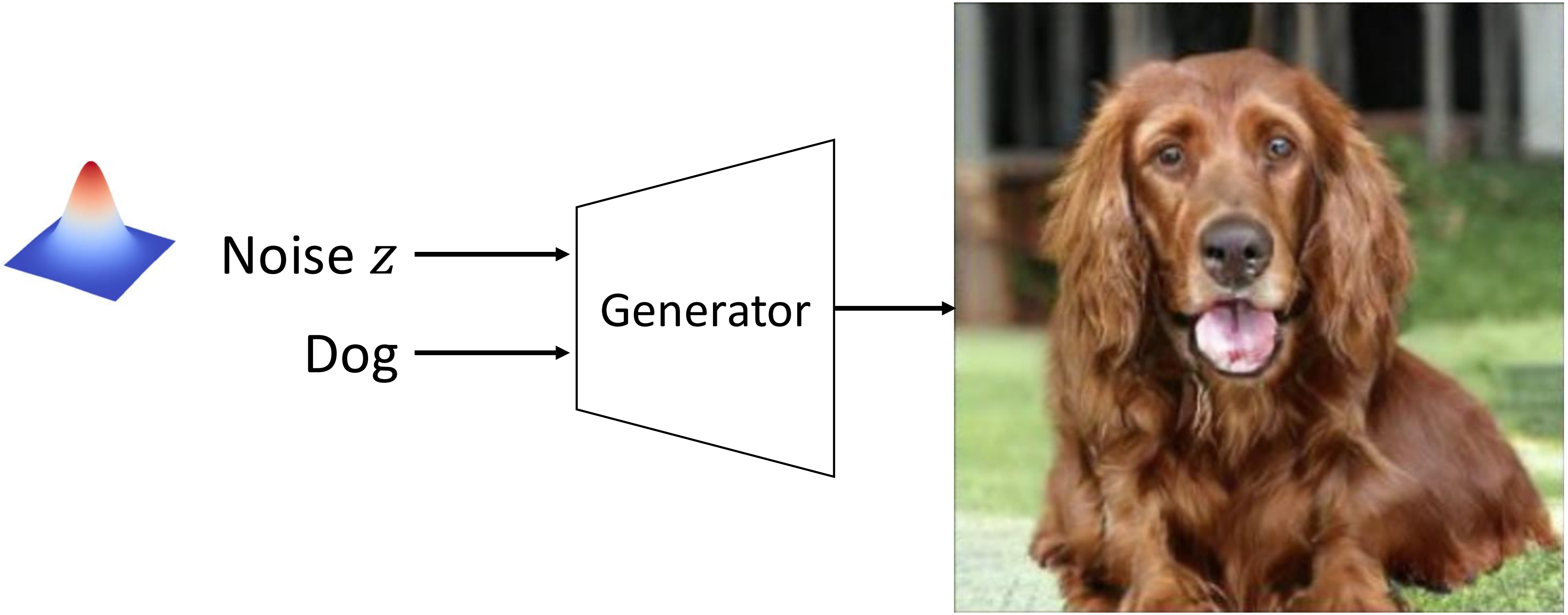
- "Improved Training of Wasserstein GANs", Gulrajani et al., NeurIPS 2017
- "Spectral Normalization for Generative Adversarial Networks", Miyato et al., ICLR 2018
- "Training Generative Adversarial Networks with Limited Data", Karras et al., NeurIPS 2020

Presentation Hint:

Understand the previous few slides and put each paper into that context

Application: Conditional GANs

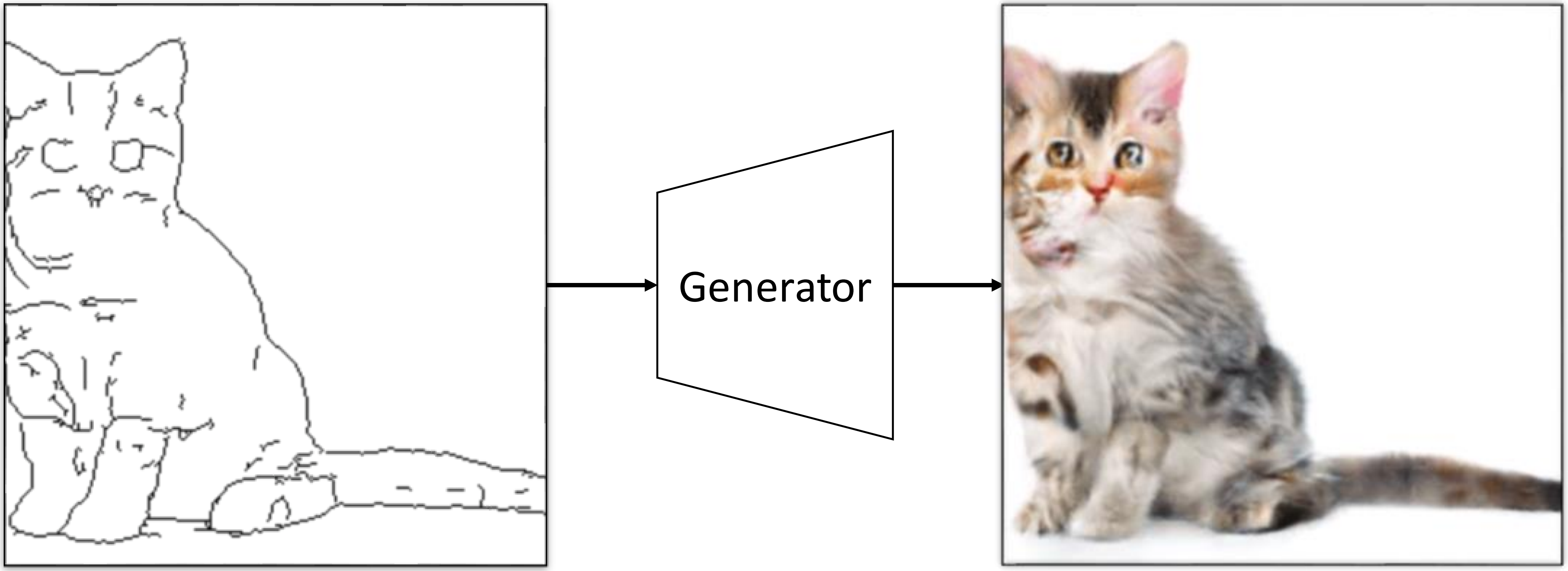
- Class-conditioned Image Generation



Brock et al., "Large Scale GAN Training for High Fidelity Natural Image Synthesis", ICLR 2019

Application: Conditional GANs

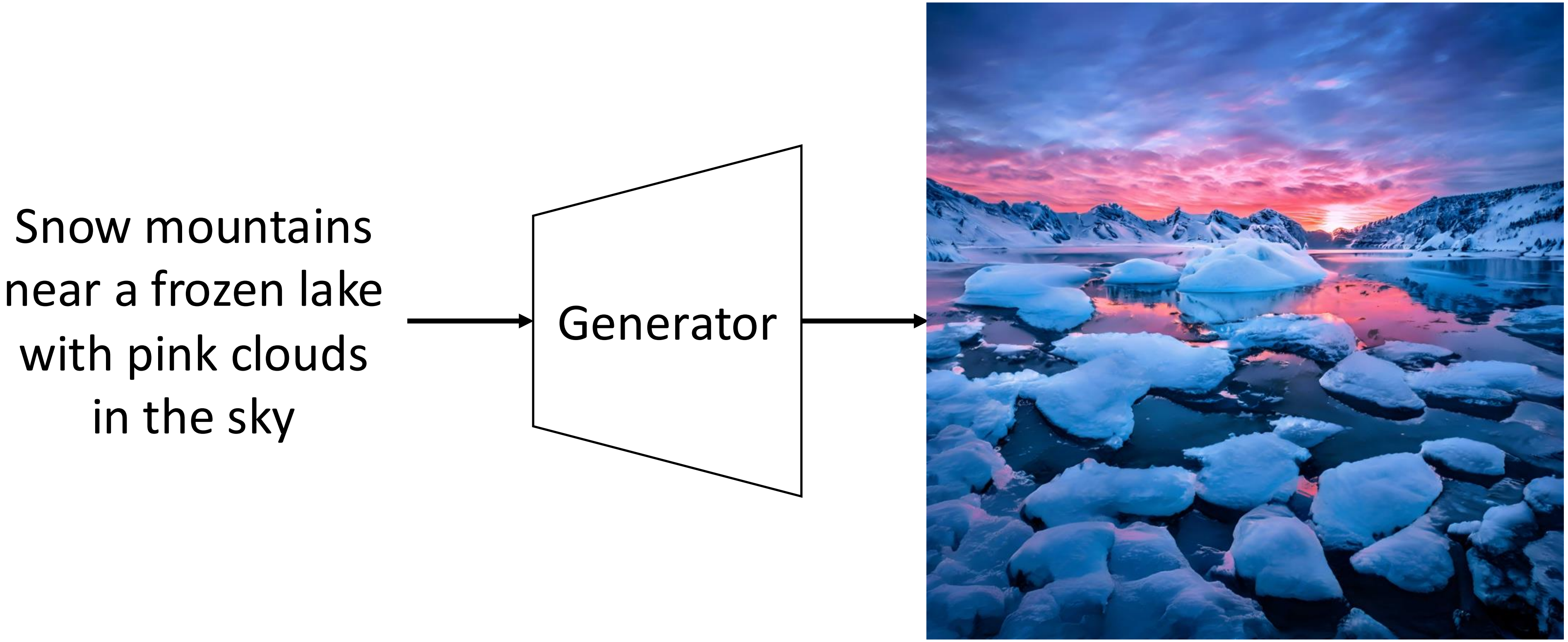
- Image-to-Image Translation



Isola et al., "Image-to-Image Translation with Conditional Adversarial Nets", CVPR 2017

Application: Conditional GANs

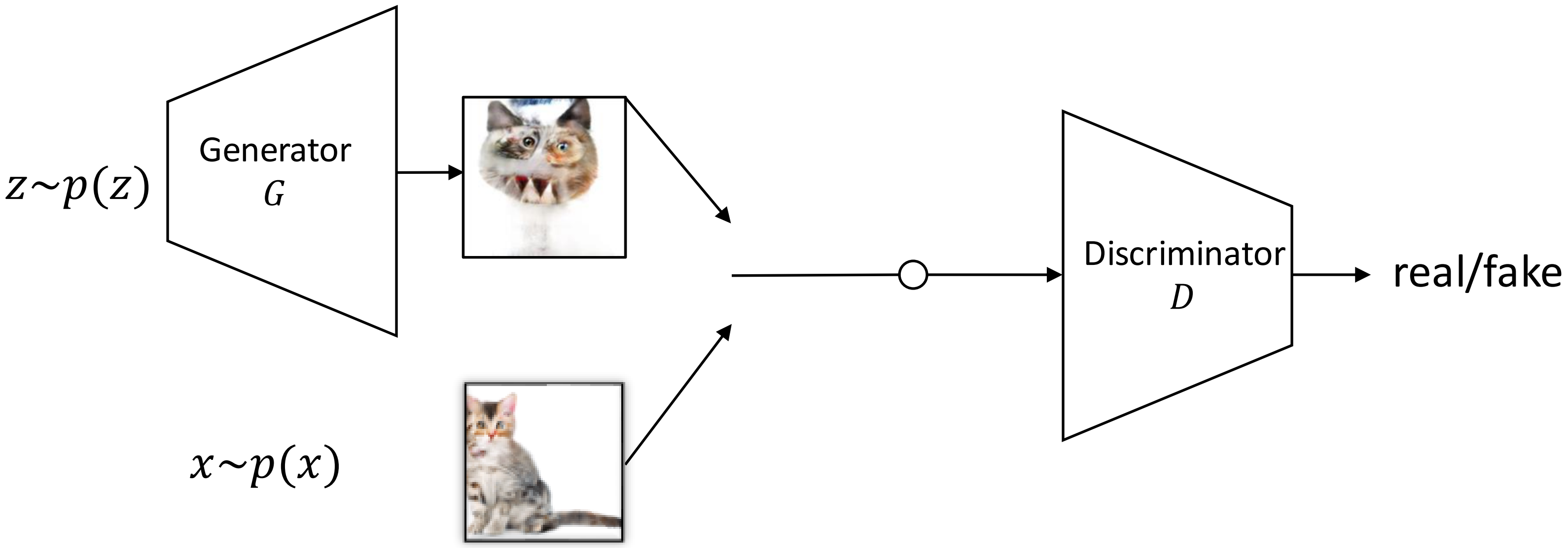
- Text-to-Image Generation



Huang et al., "Multimodal Conditional Image Synthesis with Product-of-Experts GANs", ECCV 2022

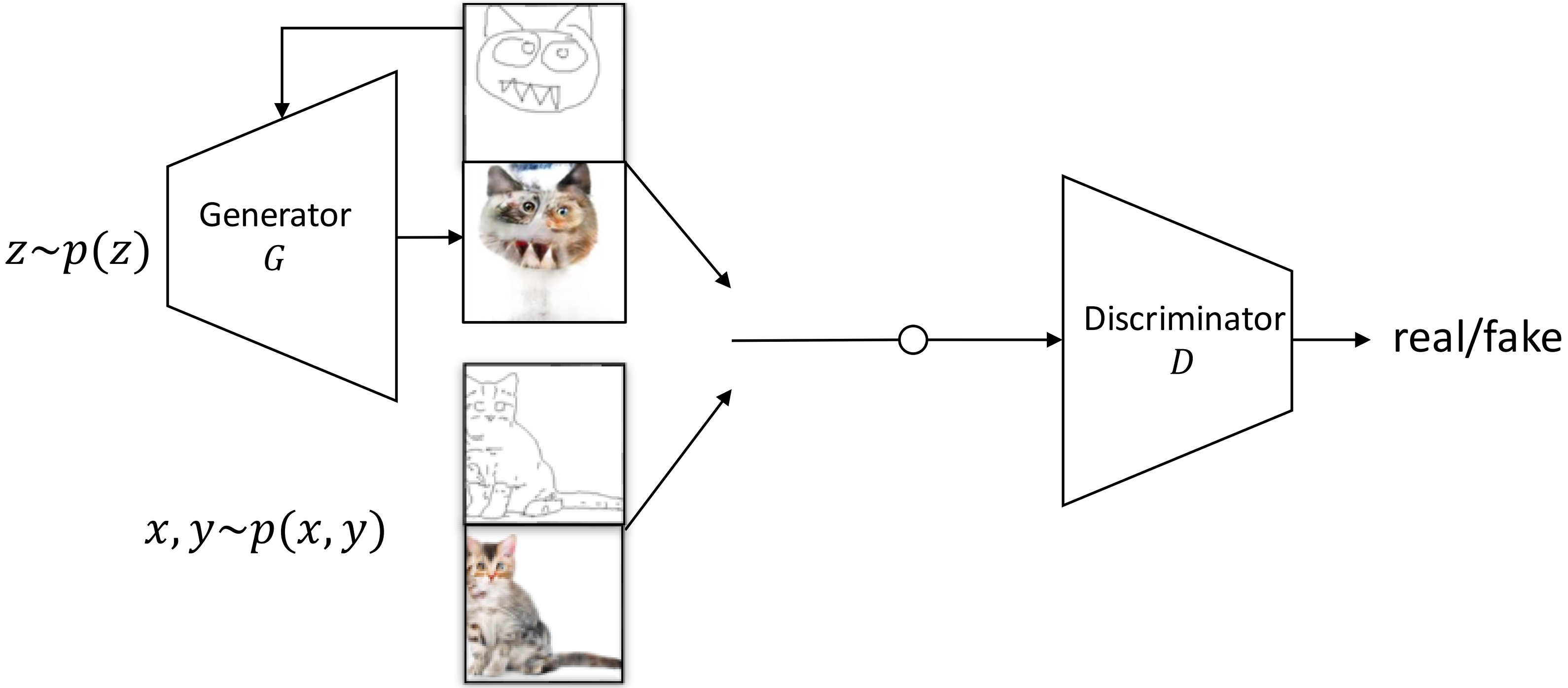
How to Condition your GANs

- It's simple! Just give your conditioning signals to both generator and discriminator as inputs.



How to Condition your GANs

- It’s simple! Just give your conditioning signals to both generator and discriminator as inputs.
- Why does it work?



5 Minute Quiz

- On Canvas
- Passcode: donkey